Name : VEDASHREE BHALERAO

Class : LY – AIA – 1

Batch : B

Roll No : 2213191

Assignment No. 1

Aim: Introduction to Keras and Tensorflow (Optional – Pytorch). Configure and use google colab and kaggle GPU **Objectives:**

1. To configure anaconda and google colab, kaggle environment
2. To Explore TF/Keras/Pytorch libraries
3. To learn to use GPU/TPU
4. To learn and understand Git **Theory:**

Keras Configuration

1. Setup Environment

2. pip install keras

3. to check whether it has installed properly: python>>import keras

Python>> print keras.__version__

Tensorflow Configuration:

1. pip install tensorflow==2.2.0

2. To verify installation: python>> import tensorflow as tf

If no error then the installation has been completed

Colaboratory Configuration

1. Setup the environment

2. Connect to the Drive

3. Upload the files using: from google.colab import files

files.upload()

Kaggle Configuration:

1. Create a Kaggle Account

2. Create an Authorization token

3. Upload on Colab using the upload code

4. Make a folder and make the Json file executable

5. Get the dataset

6. Copy the API command and run on colab

GitHub Configuration

1. pip install git

2. Set up user profile by: global user.name “name”

Global user.email “mail”

Dataset Attributes:

1. Pregnancies

2. Glucose

3. BloodPressure

4. SkinThickness

5. Insulin

6. BMI

7. DiabetesPedigreeFunction

8. Age

9. Outcome

Code:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models from
sklearn.model_selection import train_test_split from
sklearn.preprocessing import StandardScaler from
tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("/content/sample_data/diabetes.csv")

# Prepare features and target
X = df.iloc[:, 0:8] y =
df["Outcome"]

# Standardize the data scaler
= StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Split data into training and test sets
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, y,
test_size=0.1, random_state=42)

# Build the neural network model with dropout layers
model = models.Sequential([
    layers.Dense(100, activation="relu", input_shape=(X_train.shape[1],)),
layers.Dropout(0.3),
    layers.Dense(75, activation="relu"),
layers.Dropout(0.3),
    layers.Dense(50, activation="relu"),
layers.Dropout(0.3),
    layers.Dense(25, activation="relu"),
layers.Dropout(0.2),
    layers.Dense(12, activation="relu"),
layers.Dense(1, activation="sigmoid")
])

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
loss="binary_crossentropy", metrics=["accuracy"])

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the model
history = model.fit(X_train, Y_train, epochs=150, validation_data=(X_test,
Y_test), callbacks=[early_stop])

# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, Y_test, verbose=2)
print(f"Test Loss: {test_loss:.4f}") print(f"Test Accuracy:
{test_acc:.4f}")

# Plot Training Loss and Accuracy plt.figure(figsize=(12,
5))

# Plot Loss plt.subplot(1,
2, 1)
plt.plot(history.history['loss'], label='Training Loss', color='b')
plt.plot(history.history['val_loss'], label='Validation Loss',
color='orange')
plt.title('Loss over Epochs')
plt.xlabel('Epochs') plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.ylim([0, 1.0])

# Plot Accuracy plt.subplot(1,
2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy', color='g')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy',
color='red')
plt.title('Accuracy over Epochs')
plt.xlabel('Epochs') plt.ylabel('Accuracy')
```

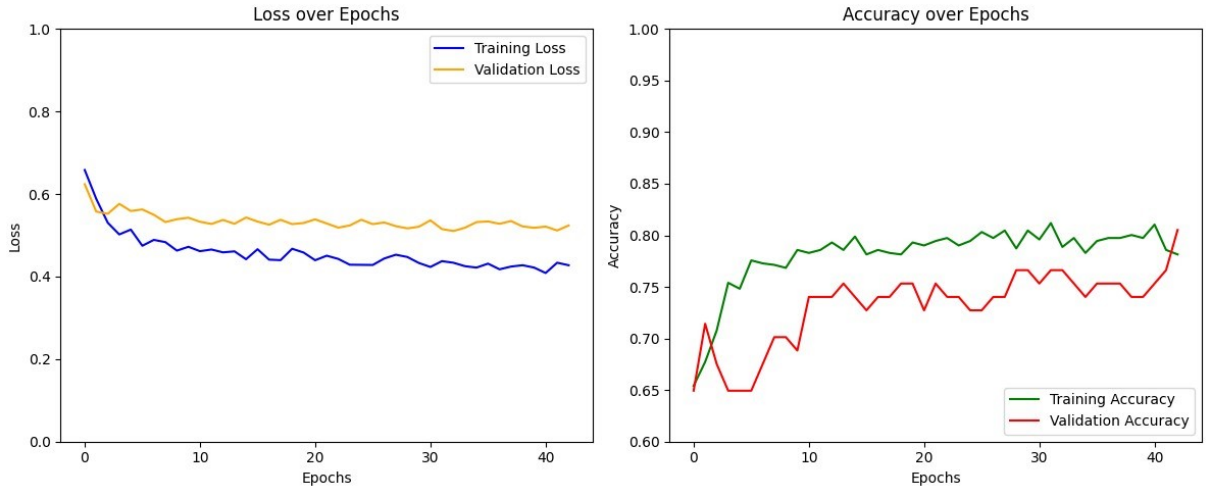
```
plt.legend(loc='lower right')
plt.ylim([0.6, 1.0])

# Show both plots
plt.tight_layout() plt.show()
```

Results：

```
Epoch 1/150
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
22/22 ━━━━━━━━━━━━━━━━━━━━ 3s 20ms/step - accuracy: 0.6205 - loss: 0.6724 - val_accuracy: 0.6494 - val_loss: 0.6238
Epoch 2/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6452 - loss: 0.6180 - val_accuracy: 0.7143 - val_loss: 0.5576
Epoch 3/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6956 - loss: 0.5491 - val_accuracy: 0.6753 - val_loss: 0.5521
Epoch 4/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7461 - loss: 0.4958 - val_accuracy: 0.6494 - val_loss: 0.5762
Epoch 5/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7499 - loss: 0.5038 - val_accuracy: 0.6494 - val_loss: 0.5590
Epoch 6/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7907 - loss: 0.4510 - val_accuracy: 0.6494 - val_loss: 0.5627
Epoch 7/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7822 - loss: 0.4490 - val_accuracy: 0.6753 - val_loss: 0.5497
Epoch 8/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7677 - loss: 0.5037 - val_accuracy: 0.7013 - val_loss: 0.5322
Epoch 9/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7784 - loss: 0.4494 - val_accuracy: 0.7013 - val_loss: 0.5391
Epoch 10/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7776 - loss: 0.4798 - val_accuracy: 0.6883 - val_loss: 0.5426

22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7825 - loss: 0.4454 - val_accuracy: 0.7532 - val_loss: 0.5336
Epoch 37/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8190 - loss: 0.3848 - val_accuracy: 0.7532 - val_loss: 0.5279
Epoch 38/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8012 - loss: 0.4402 - val_accuracy: 0.7532 - val_loss: 0.5346
Epoch 39/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8240 - loss: 0.3866 - val_accuracy: 0.7403 - val_loss: 0.5215
Epoch 40/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7850 - loss: 0.4282 - val_accuracy: 0.7403 - val_loss: 0.5182
Epoch 41/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8230 - loss: 0.3836 - val_accuracy: 0.7532 - val_loss: 0.5211
Epoch 42/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8025 - loss: 0.4217 - val_accuracy: 0.7662 - val_loss: 0.5117
Epoch 43/150
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7989 - loss: 0.4090 - val_accuracy: 0.8052 - val_loss: 0.5238
3/3 - 0s - 8ms/step - accuracy: 0.7662 - loss: 0.5105
Test Loss: 0.5105
Test Accuracy: 0.7662
```



Conclusion:

Thus we have understood the configuration steps of Google Colab, tensorflow, etc and learned to use tensorflow and create a model to predict the chance of diabetes or not.